

This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images,
please do not report the images to the
Image Problem Mailbox.



JAPANESE PATENT OFFICE

PATENT ABSTRACTS OF JAPAN

Jc918 U.S. PRO

09/748862

12/28/00

(11)Publication number: 10124331

(43)Date of publication of application: 15.05.1998

(51)Int.Cl.

G06F 9/46

(21)Application number: 08275885

(71)Applicant:

NEC ENG LTD

(22)Date of filing: 18.10.1996

(72)Inventor:

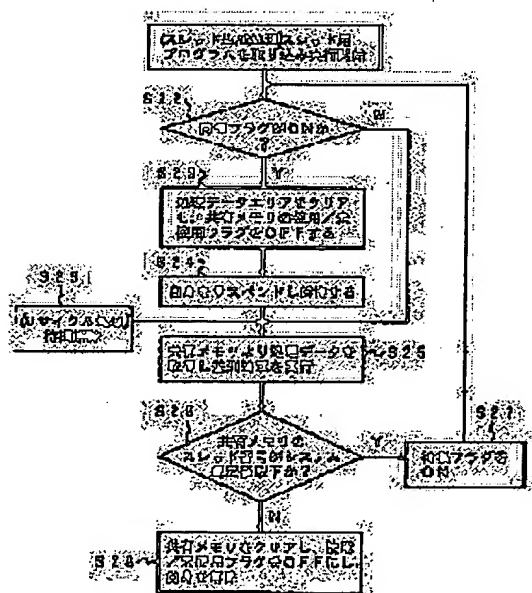
YOSHIMOTO IKUKO

(54) DATA PROCESSING METHOD

(57)Abstract:

PROBLEM TO BE SOLVED: To prevent a system from the lack of resources and the reduction of performance due to the increment of threads by reducing the number of stand-by threads up to the number of operating threads when the number of stand-by threads after parallel processing is larger than the number of threads required for system operation.

SOLUTION: A thread program is entered. While referring to a common memory, whether a stand-by flag is ON or not is checked (S22). Just after the generation of a thread, a stand-by flag corresponding to the thread is OFF. Then processing data are acquired from the common memory and processing for executing parallel processing is executed (S25). Then whether a thread number in the common memory is smaller than a regulated value indicating the number of threads at the time of normal system operation or not is checked (S26). When the thread number is larger than the regulated value, a use/unused flag, a stand-by flag and processing data in the common memory are cleared (S28).



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平10-124331

(43) 公開日 平成10年(1998) 5月15日

(51) Int.Cl.⁶

G 0 6 F 9/46

識別記号

3 4 0

F I

G 0 6 F 9/46

3 4 0 B

3 4 0 D



審査請求 未請求 請求項の数 2 O L (全 10 頁)

(21) 出願番号 特願平8-275885

(22) 出願日 平成8年(1996)10月18日

(71) 出願人 000232047

日本電気エンジニアリング株式会社
東京都港区芝浦三丁目18番21号

(72) 発明者 吉本 郁子

東京都港区芝浦三丁目18番21号 日本電気
エンジニアリング株式会社内

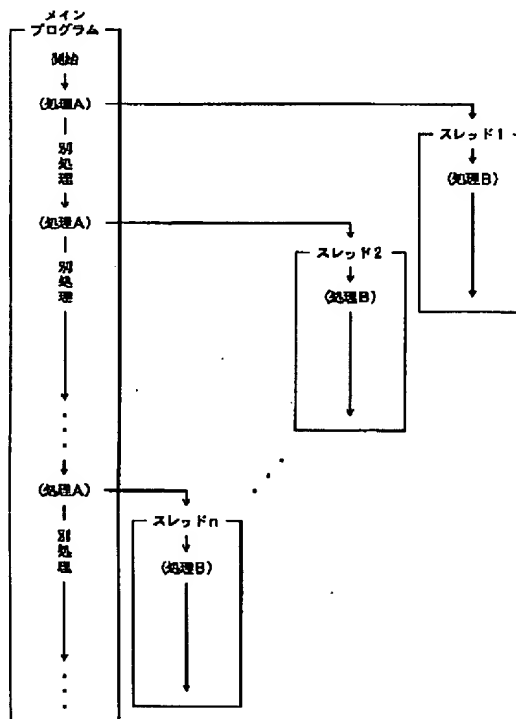
(74) 代理人 弁理士 鈴木 正剛

(54) 【発明の名称】 データ処理方法

(57) 【要約】

【課題】 スレッドの増加によるシステムの資源不足や性能低下を防止できるデータ処理方法を提供する。

【解決手段】 データ処理システム上で複数のスレッドにより並行処理を行うデータ処理方法である。並列処理の実行後に、データ処理システムのシステム運用に必要な規定値の個数のスレッドを待機状態とし、この規定値の個数を越えたスレッドを並列処理の実行後に削除する。



【特許請求の範囲】

【請求項 1】 所定のデータ処理システム上で複数のスレッドにより並行処理を行うデータ処理方法において、並列処理の実行後に、前記データ処理システムのシステム運用に必要となる規定値の個数のスレッドを並行処理が再開可能な待機状態とし、前記規定値の個数を越えたスレッドを並列処理の実行後に削除することを特徴とするデータ処理方法。

【請求項 2】 並列処理に使用される各スレッドに対して所定のスレッド番号を順次設定するとともに並行処理の実行直後に各スレッドのスレッド番号と前記規定値とを比較し、スレッド番号が前記規定値より大きいスレッドは削除し、スレッド番号が前記規定値以下であるスレッドは前記待機状態とすることを特徴とする請求項 1 記載のデータ処理方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、データ処理方法に関し、特に、データ処理システムにおいてスレッドの生成や削除などを管理する方法に関する。

【0002】

【従来の技術】例えばコンピュータを用いたデータ処理システムにおいて、メインプログラムの実行中に、並行処理、例えば、メインプログラムの処理と並行して特定の関数を実行する場合がある。上記並行処理は、例えばスレッドを用いて行われる。このスレッドを用いる場合、一度生成したスレッドを削除することなく再利用することで、スレッドの生成時間を短縮・削減し、システムにおける処理の高速化を図ることも試みられている。このような技術については、例えば特開平 4-23137 号や特開平 6-318191 号公報などに開示されている。このようなスレッドを用いた従来のデータ処理の一例を、図 6～図 8 を用いて以下に説明する。

【0003】まず、図 6 において、リサイクル可能なスレッドの個数をチェックする（ステップ S 31）。このような個数チェックは、図 8 のリサイクルスレッドカウンタ G のカウンタ値から判別できる。そして、この個数が 0 の場合には、リサイクル可能なスレッドが存在しないことから、ステップ S 36 以降の処理に移行し、スレッドの生成処理を行う。一方、上記の個数が 0 でなく、リサイクル可能なスレッドが存在する場合は、スレッド管理テーブルの先頭から各スレッドの使用／未使用フラグ B を順次チェックし、これらの使用／未使用フラグの状態をチェックする（ステップ S 32）。

【0004】そして、使用／未使用フラグがオフで未使用のスレッドがある場合は、そのスレッドの使用／未使用フラグをオン、つまり使用中にし（ステップ S 33）、また、図 8 の処理データ D に処理に必要なデータをセット（ステップ S 34）する。さらに、図 8 のスレッド番号（ハンドル値）A に基づいてスレッドの再開処

理を行う（ステップ S 35）。

【0005】一方、使用／未使用フラグがオフで未使用のスレッドが存在しない場合は、テーブルの先頭から上記のリサイクル可能なスレッド個数分だけ飛ばした位置にある、スレッド管理テーブル上の空きエリアを確保し、この空きエリアにおける使用／未使用フラグを使用中とする（ステップ S 36）。また、この空きエリアに、スレッド番号として、スレッド管理テーブルの先頭から何番目のエリアであったかをセットする（ステップ S 37）。そして、上記のようにスレッドが生成され、リサイクル可能なスレッド数が増えたことから、図 8 のリサイクルスレッドカウンタ G を +1 する（ステップ S 38）。

【0006】次に、上記でセットしたエリアをスレッドとメインプログラムとの共用メモリに指定するため、スレッド生成時において指定する共用メモリアドレスとして、データをセットしたスレッド管理テーブル内のエリアのアドレスを、図 8 の E のアドレスに割り当てる（ステップ S 39）。これにより、スレッドの生成処理が完了し、その後、ステップ S 34 とステップ S 35 の処理を行ってスレッドの実行の指示をする。

【0007】ここで、共用メモリは、スレッドが参照して動作するためのものである。そして、スレッドを必要とする場合には、管理テーブルから未使用のスレッドを検索し、未使用スレッドに該当する管理テーブル内のエリアの使用／未使用フラグを使用中とし、スレッドの動作に必要なデータを所定のエリア内にセットし、その後、親スレッドが動作のトリガとして、そのスレッドのリジウムを行い、処理動作を指示する。また、スレッドは、割り当てられた親スレッドとの共用メモリアreaから必要なデータを取得し、動作を開始する。さらに、この共用メモリを書き替えることで、親スレッドはスレッドに対して所定の動作を指示し実行させることができる。

【0008】また、図 7 は、スレッドにおける並行処理の手順を示したもので、スレッドは、スレッド用プログラムを取り込んで並行処理の実行を開始する。まず、スレッドは、その生成後すぐに、それ自身をサスペンドして待機状態とする（ステップ S 42）。なお、サスペンドとは、スレッドを中断して再開可能な待機状態にすることをいう。そして、スレッドは、上記のような再開の指示があった場合、共用メモリ F から処理データを取得し、並行処理を実行する（ステップ S 43）。さらに、並行処理実行後は、処理データエリアをクリアし、使用／未使用フラグ B を未使用にセットし、再び待機状態になる（ステップ S 42）。

【0009】以上のようにして、全てのスレッドがリサイクル可能なスレッドとなり、また、使用されないときには常に待機状態となる。さらに、スレッドの削除は、システムのシャットダウン時までには行われない。

【0010】

【発明が解決しようとする課題】ところで、上記のようなデータ処理システムにおいてシステムに負担がかかった場合には、当然のことながら、スレッドの個数が通常のシステム運用時の個数よりも増大する。そして、上記従来の方法の場合、スレッドの個数は、例えば、図9のように、通常運用時の個数よりも増加する。

【0011】ところが、この従来は、スレッドの個数が増大した後に通常の運用に戻った場合、図9のようにスレッドの個数が増大したままとなる。このため、使用されない複数のスレッドが待機状態となって、システムの資源を無駄に使用し続けるという問題があった。

【0012】また、スレッドは、待機状態であっても、非常に少ない値ではあるがCPU時間を消費する。よって、上記従来手法では、システムに負荷がかかった後に通常の運用に戻った際、待機状態のスレッドによるCPU時間の消費量が増加してしまう。このため、通常の運用に戻った際の運転時におけるシステムの処理性能の低下を招くという問題もあった。

【0013】本発明は、上記のようなスレッドの増加によるシステムの資源不足や性能低下を防止できる、改良されたデータ処理方法を提供することを課題としている。

【0014】

【課題を解決するための手段】本発明のデータ処理方法は、所定のデータ処理システム上で複数のスレッドにより並行処理を行うデータ処理方法において、並列処理の実行後に、前記データ処理システムのシステム運用に必要となる規定値の個数のスレッドを並行処理が再開可能な待機状態とし、前記規定値の個数を越えたスレッドを並列処理の実行後に削除することを特徴とする。

【0015】ここで、システム運用に必要となる規定値の個数とは、システム上における通常の運用において使用する任意の個数ないし規定値を意味する。このような規定値は、システムに応じて変化するものであり、システムに応じた任意の値とすれば良い。

【0016】つまり、本発明では、並行処理後における待機スレッドの数がシステム運用に必要なスレッド（運用スレッド）の数よりも多い場合には、待機スレッドの数を運用スレッドの数まで削減するようにしたものである。

【0017】上記のような並行処理実行後におけるスレッドの削除を行うため、例えば、並列処理に使用される各スレッドに対して所定のスレッド番号を順次設定するとともに並行処理の実行直後に各スレッドのスレッド番号と前記規定値とを比較し、スレッド番号が前記規定値より大きいスレッドは削除し、スレッド番号が前記規定値以下であるスレッドは前記待機状態とする、手順が用いられる。

【0018】

【発明の実施の形態】以下に、本発明のデータ処理方法の実施の形態を説明する。図1は、本発明が適用されるデータ処理システムにおける処理の概要を示したものである。そして、このメインプログラムの実行中において、複数のスレッドにより並行処理を行う必要が生じた場合には、メインプログラムにおいては以下に説明する処理Aが、また各スレッド（スレッド1～スレッドn）においては以下に説明する処理Bがそれぞれ行われる。

【0019】まず、データ処理システムのメインプログラムにおける処理Aについて、つまり、メインプログラムにおけるスレッド管理処理について、図2を参照して説明する。この処理は、次の（1）～（4）のように行われる。

（1） 並行処理が必要となった時は、リサイクルスレッドカウントが0であるかをチェックするとともに、リサイクルスレッドが存在するかどうかをチェックする（ステップS1）。

【0020】なお、リサイクルスレッドカウントは、例えば、図4に符号Gとして例示したように、データ処理システムの所定のスレッド管理テーブルに設定されるもので、スレッド管理テーブル内におけるリサイクル可能なスレッドの数を示すものである。また、このスレッド管理テーブルには、このテーブルが管理しているスレッドの数に対応するだけの共用メモリFがあり、各共用メモリFは、スレッド番号（ハンドル値）A、使用／未使用フラグB、待機フラグC、処理データDがそれぞれ設定されている。

【0021】ここで、スレッド番号Aは、スレッド管理テーブルにおいて管理されるスレッドに対して順次設定される番号を示したものである。また、使用／未使用フラグBは、各スレッドの使用状態を示すためのものである。さらに、待機フラグCは、そのスレッドが生成直後ではなくてリサイクル可能なスレッドであることを示すためのものである。さらに、処理データDは、メインプログラムから並行処理に必要なデータをスレッドへ提供するためのものである。

【0022】そして、リサイクルカウントが0であった場合、つまり、リサイクルスレッドが存在しない場合には、以下の（4）の処理に進み、スレッドの生成処理を行う。また、リサイクルカウントが0でない場合には、以下の（2）の処理に進み、利用可能なリサイクルスレッドが存在するかどうかのチェックを行う。

【0023】（2） リサイクルスレッドカウントに対応する数のスレッド管理テーブル内の使用／未使用フラグBをチェックする（ステップS2）。そして、未使用のリサイクルスレッドがない場合は、下記（4）のスレッド生成処理に進む。また、未使用のリサイクルスレッドがある場合は、以下の（3）のリサイクルスレッドを利用した処理へ進む。

【0024】（3） リサイクルスレッドの利用は、ま

ず、使用しようとしているスレッドの共用メモリに指定される管理テーブルの待機フラグCをオンにする（ステップS3）。次に、使用／未使用フラグBをオンにして使用中とし（ステップS4）、並行処理に必要なデータを処理データDにセットし（ステップS5）、スレッドを再開させる（ステップS6）。すると、スレッドは、ステップS29より待機状態から解かれて再開され、図3に示したリサイクル処理（S29）が行われる。

【0025】（4）スレッドの生成処理は次のように行われる。まず、使用しようとしているスレッドの共用メモリFの待機フラグCをオフにする（ステップS7）。次に、その共用メモリFにおける使用／未使用フラグBを使用中とする（ステップS8）。さらに、スレッド管理テーブルの共有メモリFのスレッド番号Aをスレッド番号Aを設定し、また並行処理に必要なデータを処理データDにセットする（ステップS9）。次に、今後このスレッドをリサイクルスレッドとして利用するかどうかの判断として、スレッド番号Aが、データ処理システムにおいて通常使用するスレッド数（規定値）より大きいかをチェックし（ステップS10）、小さい場合はリサイクルスレッドとして利用することから、スレッド管理テーブルのリサイクルスレッドカウンタGを+1する（ステップS11）。そして、スレッド管理テーブルの上記で設定およびセットを行ったエリアの共用メモリFをメインプログラムとスレッドとの共用メモリに指定して、スレッドの生成を行う（ステップS12）。なお、このようにして生成されたスレッドは、図3のステップS21から処理を開始する。

【0026】次に、各スレッド内での処理を、図3により説明する。この処理は、スレッドの生成直後における処理と、スレッドのリサイクル時における処理の2つに分かれる。

【0027】まず、スレッドの生成直後におけるスレッド内での処理は次の通りである。最初に、スレッド生成処理として、スレッド用プログラムが取り込まれ、スレッド生成処理が開始される（ステップS21）。次いで、共用メモリFを参照して、待機フラグCがオンであるかどうかをチェックする（ステップS22）。ここで、スレッド生成直後においてはそのスレッドに対応する待機フラグCはオフになっている。そして、次に、共用メモリFから処理データDを取得し、並行処理を実行する処理が行われる（ステップS25）。

【0028】次いで、共用メモリFのスレッド番号Aが通常のシステム運用時におけるスレッド数の規定値より小さいかどうかをチェックする（ステップS26）。そして、スレッド番号Aが規定値よりも大きい場合は、その共用メモリFにおける使用／未使用フラグB、待機フラグC、並びに処理データDがそれぞれクリアされ、終了する（ステップS28）。なお、使用／未使用フラグBをクリアすることは、この共用メモリFを未使用にす

ることを意味する。また、処理データDをクリアすることは、そのスレッドを削除することを意味する。

【0029】また、スレッド番号が規定値よりも小さいか、または同じであった場合は、その共用メモリFにおける待機フラグCをオンにして（ステップS27）、ステップS22に処理が移行する。そして、ステップS22では、待機フラグCがオンであるので、リサイクル処理であるステップS23、つまり、処理データDのクリアと使用／未使用フラグBを未使用にセットする処理が行われる。次いで、ステップS24において、そのスレッドはそれ自身をサスペンドして待機状態となる。

【0030】次に、スレッドのリサイクル時における処理を説明する。この処理は、次の（11）～（12）のように行われる。

【0031】（11）すなわち、リサイクル時においては、スレッドはステップS29のサスペンドして待機している状態から始まる。そして、メインプログラムから待機が解除されるので、共用メモリFから処理データDを取得し、並行処理を実行する（ステップS25）。

【0032】（12）次いで、共用メモリFのスレッド番号Aが通常のシステム運用時におけるスレッド数の規定値より小さいかどうかをチェックする（ステップS26）。そして、スレッド番号Aがこの規定値よりも以下であるので、共用メモリFの待機フラグCをオンにして（ステップS27）、ステップS22に処理を移す。そして、ステップS22において、待機フラグCがオンであるので、リサイクル処理であるステップS23とステップS24が行われる。

【0033】以上のように構成されるこの実施の形態のデータ処理方法では、システムに負荷が発生して一時的にスレッド個数が図5の1のように増大しても、並列処理の終了後は、図5で11のように通常のシステム運用に必要なスレッド以外は削除され、これら必要なスレッドのみ待機状態となりリサイクル可能なスレッドとなる。

【0034】このため、従来のように必要のないスレッドが待機状態であるために使用されるCPU時間の消費が削減されて、負荷発生後におけるシステムの性能低下が防止できる。また、無駄なスレッドが生成され存在し続けることで使用されるシステム資源の無駄使いがなくなる、システムの有効な資源確保が行える。さらに、システムで正常時に必要なスレッドの数のスレッドを待機状態でリサイクルする構成であるので、その分のスレッドの生成や削除にかかる時間が削減されてシステムの性能向上が図れる。

【0035】

【発明の効果】以上の説明から明らかなように、本発明によれば、スレッドの増加によるシステムの資源不足や性能低下が抑制される効果がある。

【図面の簡単な説明】

【図1】本発明が適用されるデータ処理システムにおける処理の概要を示した説明図。

【図2】実施の形態のデータ処理システムにおけるスレッド管理処理を示したフローチャート。

【図3】実施の形態のデータ処理システムにおける各スレッド内での処理を示したフローチャート。

【図4】実施の形態のデータ処理システムにおけるスレッド管理テーブルの説明図。

【図5】実施の形態のデータ処理システムにおけるスレッド数の推移を示した説明図。

【図6】従来のデータ処理システムにおける処理を説明したフローチャート。

【図7】従来のデータ処理システムにおける処理を説明

したフローチャート。

【図8】従来のデータ処理システムにおけるスレッド管理テーブルの説明図。

【図9】従来のデータ処理システムにおけるスレッド数の推移の説明図。

【符号の説明】

A スレッド番号

B 使用／未使用フラグ

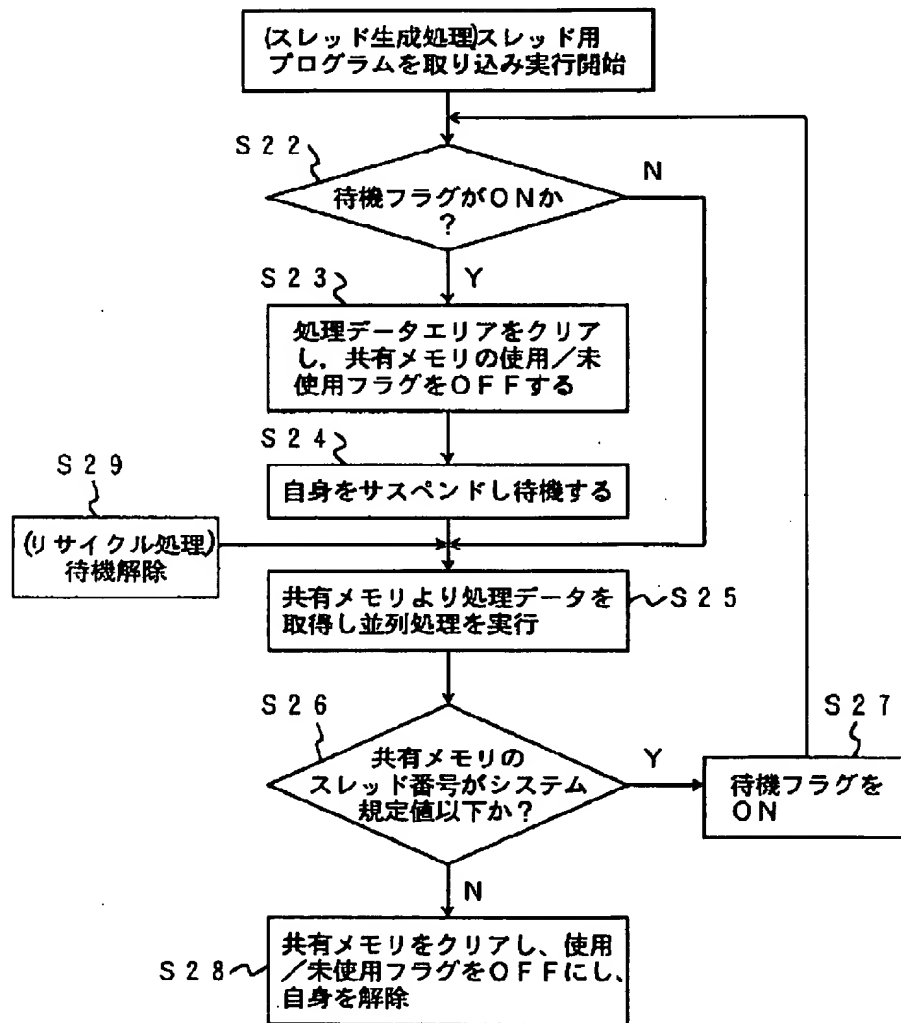
C 待機フラグ

10 D 処理データ

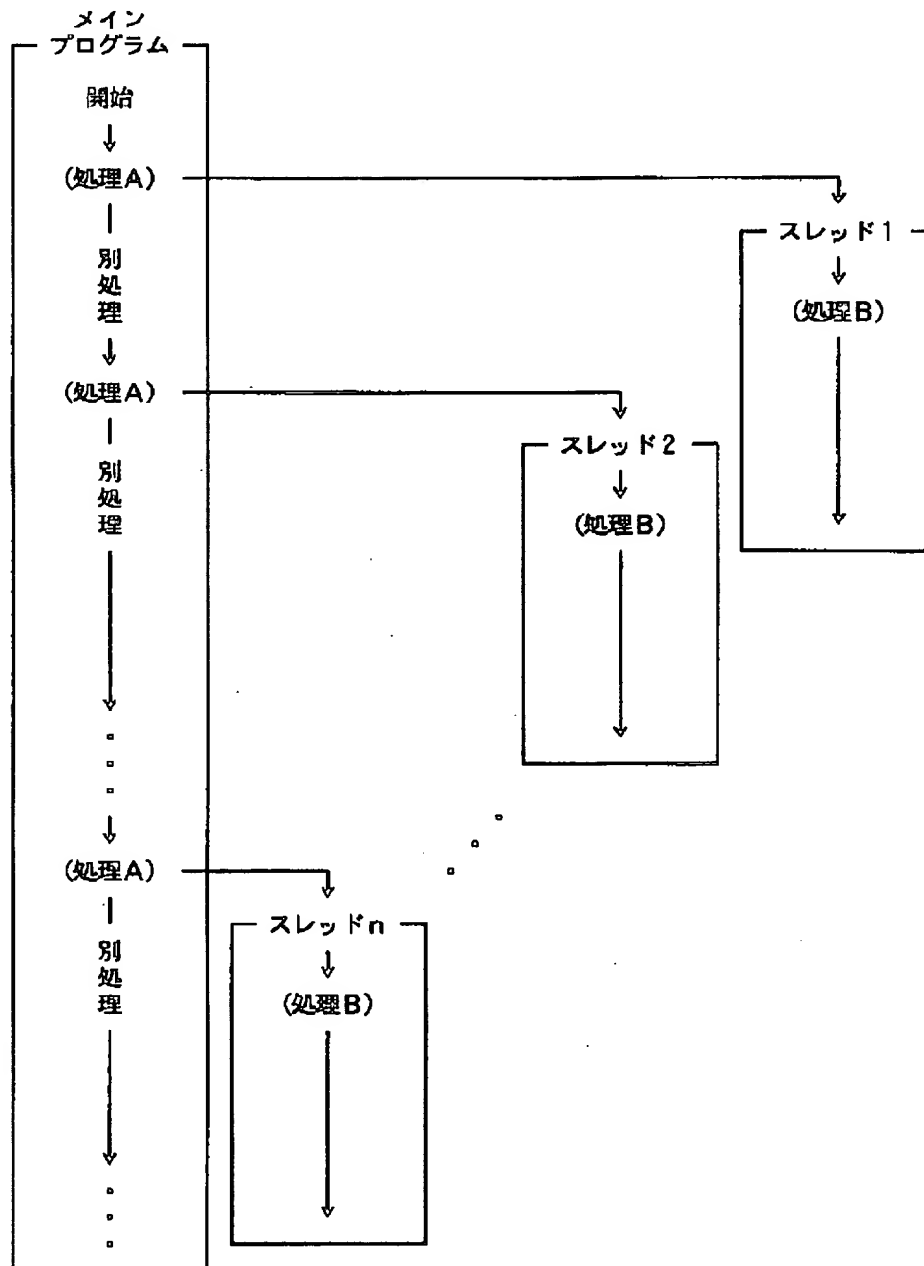
G リサイクルスレッドカウンタ

F 共用メモリ

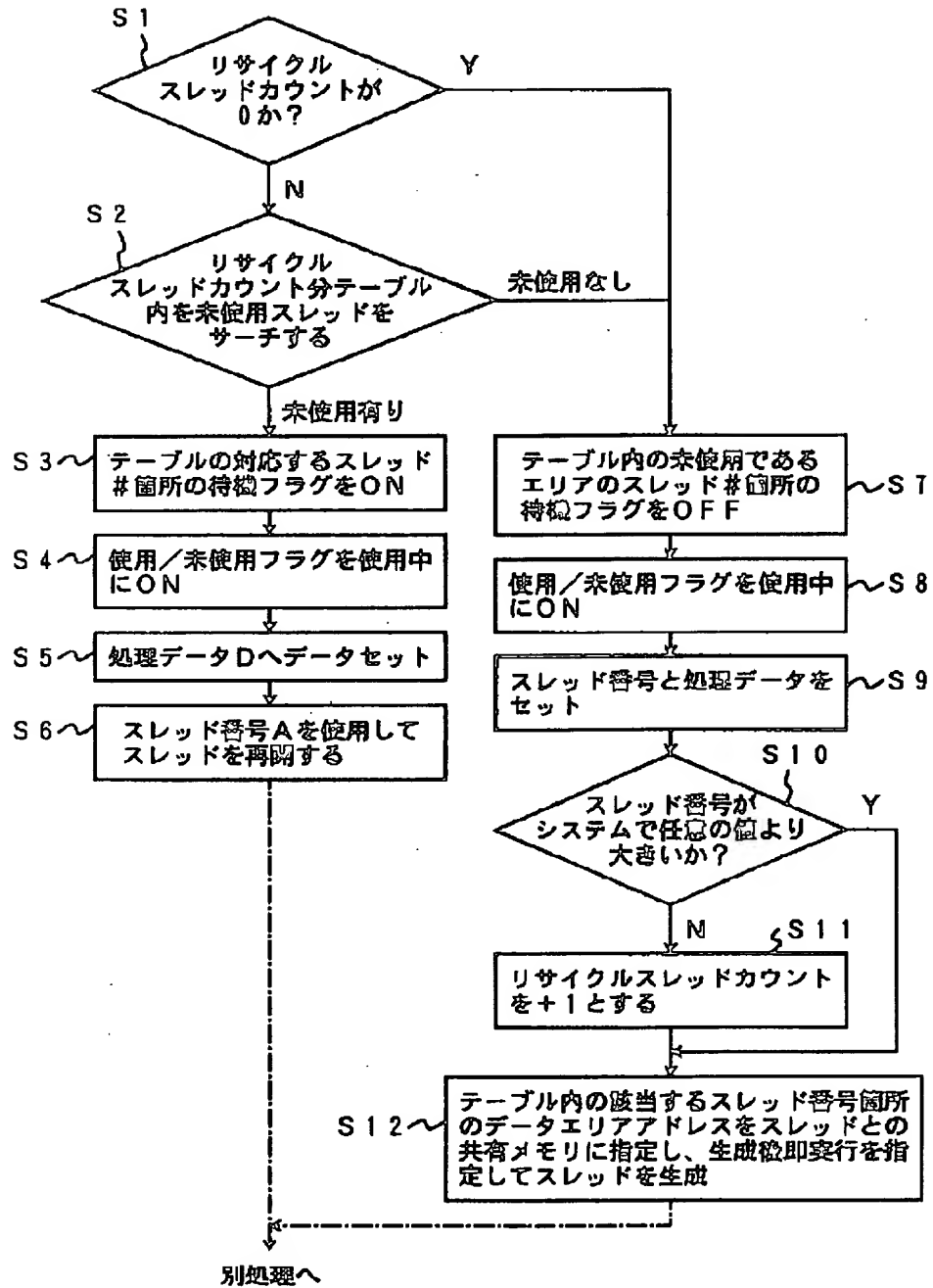
【図3】



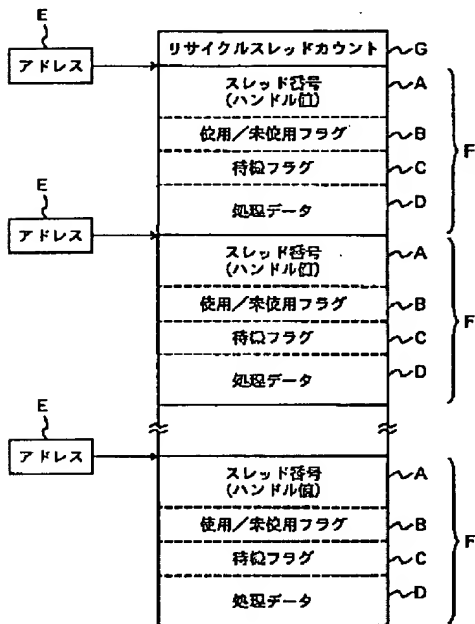
【図1】



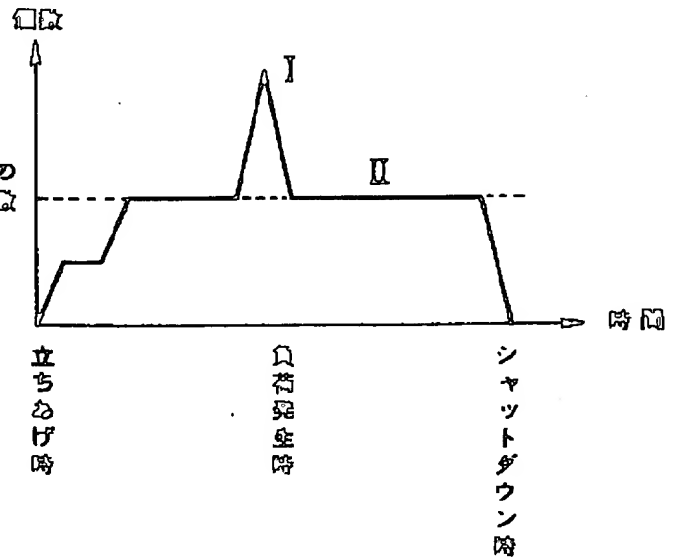
【図2】



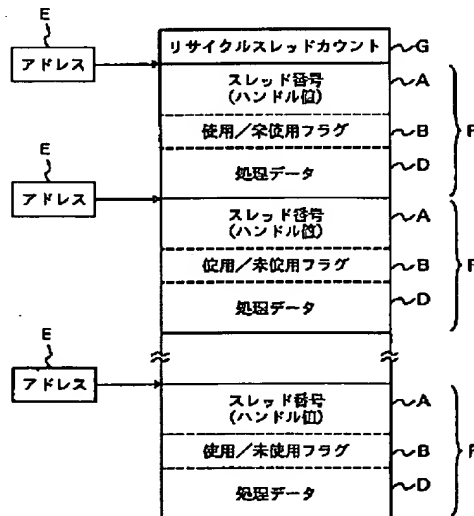
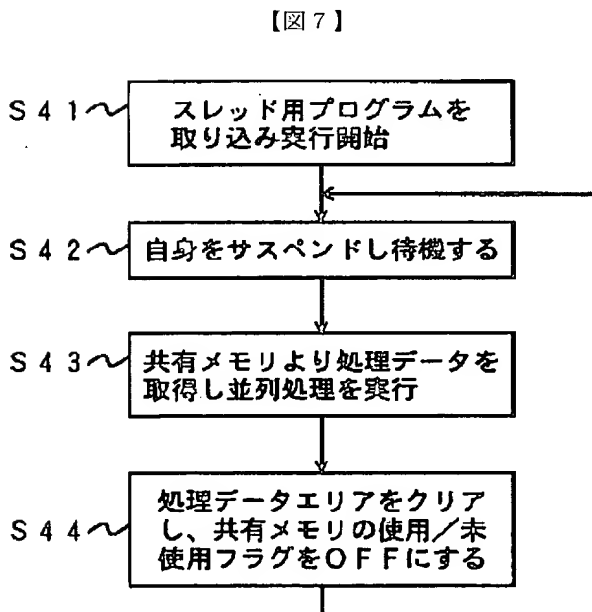
【図4】



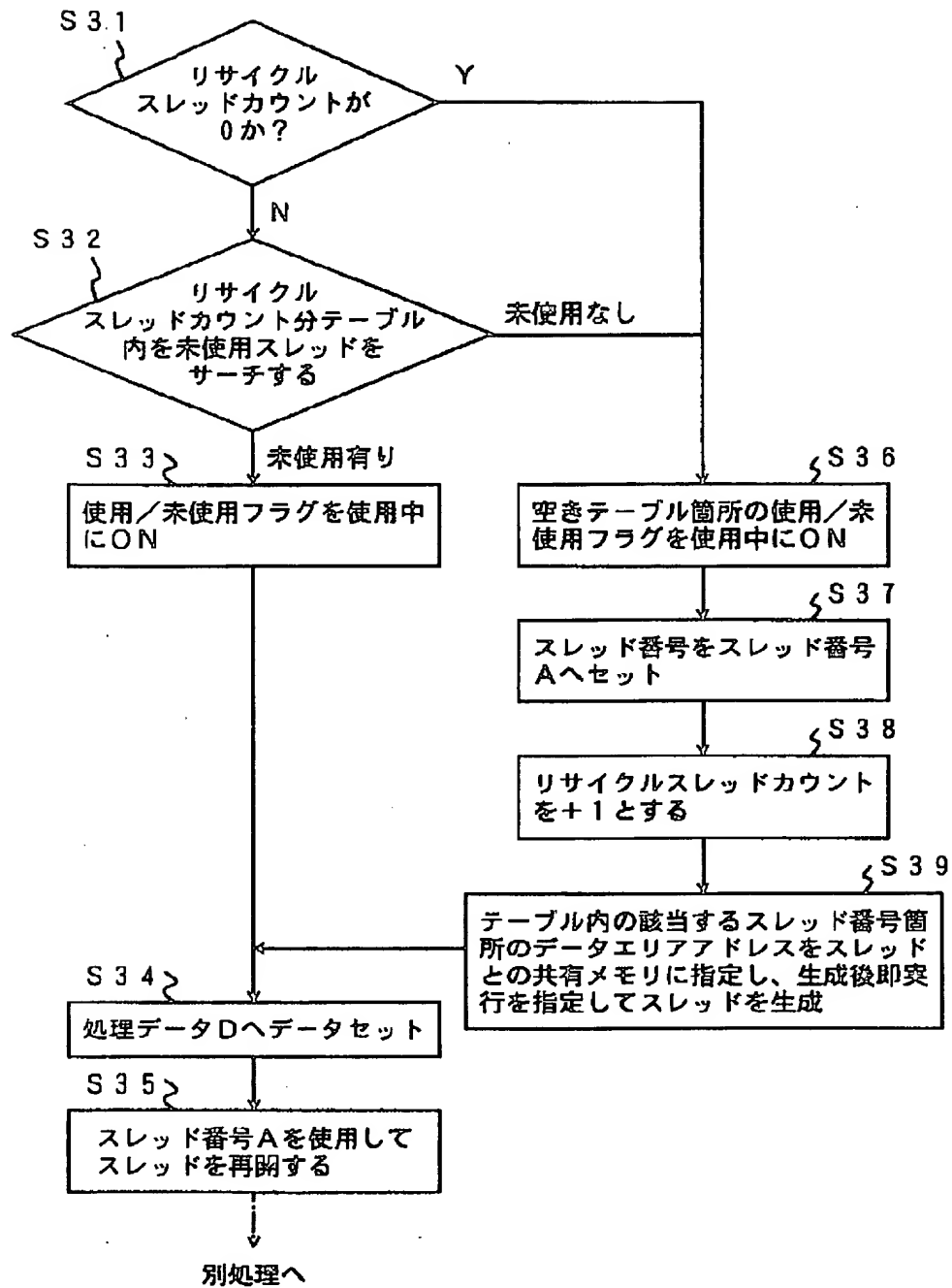
【図5】



【図8】



【図6】



【図 9】

